3D Visual Data Mining: goals and experiences

Bøhlen, Michael Hanspeter; Bukauskas, Linas; Eriksen, Poul Svante; Lauritzen, Steffen Lilholt; Mazeika, Arturas; Musaeus, Peter; Mylov, Peer

Published in:
Computational Statistics & Data Analysis

Publication date:
2003

Document Version
Early version, also known as pre-print

3D Visual Data Mining—Goals and Experiences

Michael Böhlen* Linas Bukauskas Poul Svante Eriksen
Steffen Lilholt Lauritzen Artūras Mažeika Peter Musaeus Peer Mylov

Aalborg University, Aalborg, Denmark

Abstract

The visual exploration of large databases raises a number of unresolved inference problems and calls for new interaction patterns between multiple disciplines—both at the conceptual and technical level. We present an approach that is based on the interaction of four disciplines: database systems, statistical analyses, perceptual and cognitive psychology, and scientific visualization. At the conceptual level we offer perceptual and cognitive insights to guide the information visualization process. We then choose cluster surfaces to exemplify the data mining process, to discuss the tasks involved, and to work out the interaction patterns.

Key words: visual data mining, immersive data exploration, observer relative data extraction, perception of space and objects, nested density surfaces

1 Introduction

The last years witnessed a continued growth of the amount of data being stored by companies in the hope of gaining competitive advantages. The advances in data gathering have created an increasing need for computational and graphical techniques to aid in data analysis. Data mining builds on theories and techniques from many fields, including pattern recognition, high performance computing, data visualization, and online analytical processing. This paper describes an interdisciplinary approach to data mining by combining expertise in statistics, database systems, visualization, and cognitive psychology with facilities for advanced 3D visualization. We investigate an immersive 3D approach that is faithful to our perception of the real world and is a good match for the interpretative power of our visual system. We extend current visualization and database system technologies to create the basis for a strong coupling between data and vision. For example, to use the full range of a 6-sided Cave, data must be provided 50 times per second. This calls for new interaction patterns and access structures to produce the data with the required frequency.

Data mining aims to discover something new from the facts recorded in a database. From a statistical point of view, databases are usually uncontrolled convenience samples; hence data mining poses a collection of challenging inference problems, raising many issues, some well studied and others unexplored or unsettled [9]. Graphical illustrations of the data, plots and graphs, which generally are convenient aids for statisticians then come to play an essential role as a tool

*Correspondence to: M. Böhlen, Department of Computer Science, Fredrik Bajers Vej 7E, DK-9220 Aalborg, Denmark, boehlen@cs.auc.dk.

for analysis and exploration of inherent structure in data. This enables visual cognition to play a major part. Statistics traditionally employs 2D techniques, including geometric, icon-based, pixel-oriented, hierarchical, and graph-based methods. It exploits the possibility to "look at data" on a computer screen with several graphical windows, access to a statistical toolbox, a keyboard and a mouse [15].

The cooperation of the four key components of an advanced data mining system is illustrated in Figure 1. Database technology selects and provides the desired data (sub)sets from the large database to be subjected to appropriate statistical processing. Using expertise from perceptual and cognitive psychology, structures amenable to visual perception are created. The observer has various interface controls, which allow feedback to the system for controlling the current data selection, the statistical and visual processing, and his own position and orientation relative to the data visualized. The lines at the bottom illustrate the feedback the data analyst may trigger. Depending on the request different processes will be activated. The line at the top illustrates that the disciplines are tightly intertwined. For example, to extract a minimal subset of the data, knowledge about statistical and visual processing as well as perceptual psychology is required. In this paper we focus on observer relative data extraction, the estimation of 3D probability density functions, and the potential of nested surfaces to detect more and less pronounced structures in the data. The scientific visualization part, including the rendering, and aspects of high-dimensional data visualization are discussed elsewhere [20]. Throughout, we provide detailed algorithms and empirical evaluations that relate the different components and contribute to a comprehensive understanding of a visual data mining system.

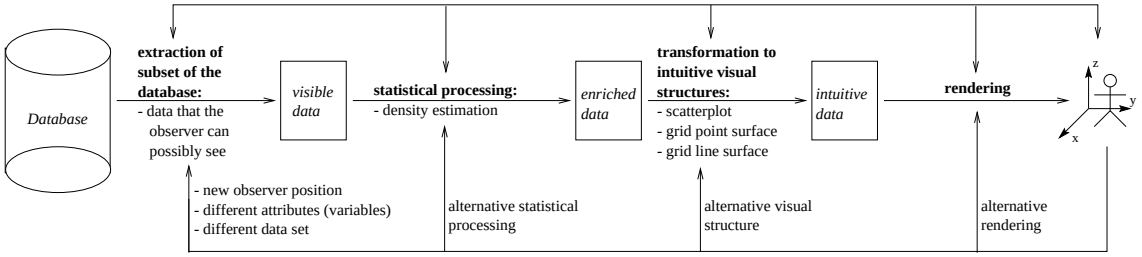


Figure 1: Data Flow and Interactions

To allow for a fine-grained interaction between the disciplines we develop database access structures that support the extraction of data that the observer can possibly see. The direct visualization of the data is not particularly attractive because the scatterplot, which is probably the most effective method for finding patterns in 2D data, is less intriguing in 3D. One of the main problems of 3D scatterplots is that the shape of point clouds cannot easily be seen even when stereo and motion cues are used [32, p. 304]. We add shape information by estimating the density of the data, and adding objects that form 2D surfaces in a 3D space and emphasize density levels. Nested surfaces are fairly robust with respect to the number of observations and they support the detection of multiple structures because they equalize the more and less pronounced structures in the data.

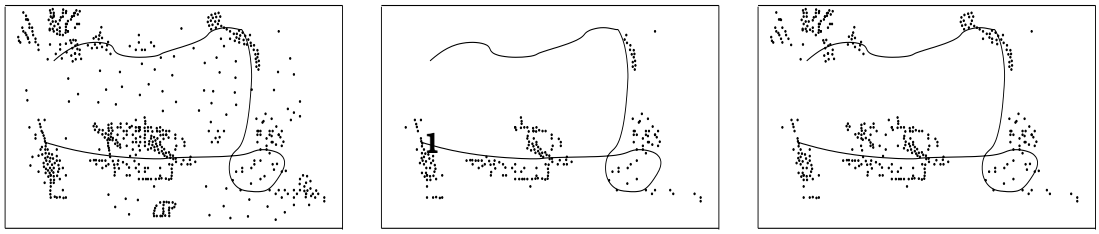
The paper proceeds as follows. Section 2 discusses the interaction with the database. Section 3 gives algorithms for estimating 3D probability density functions. Section 4 reflects about space and objects from a perceptual and cognitive perspective. The computation of nested surfaces is discussed in Section 5. Section 6 gives experimental results. Finally, related work is discussed in Section 7, and Section 8 summarizes the paper and points to future work.

2 Interacting with the Database

It is frequently argued that the increased amount of information being gathered and the advances in hard- and software makes it attractive, if not necessary, to use advanced visualization systems to analyze data. Nevertheless, two main components of such a system, a database sub-system to manage the data and a visualization sub-system to render the data, are often loosely coupled. Typically, the visualization component extracts all or much of the data from the database (or a file) and organizes it in a scene tree, which is then used to visualize the data.

This section proposes extensions to the database system part that permit a more tightly coupled architecture. We investigate *observer relative data extraction* (ORDE), which makes it possible to retrieve a controlled super- or subset of the data that the observer can see. Whether or not the observer can see an object depends on the visibility factor of the object. The visibility factor, in turn, depends on the properties of the object and the distance between the observer and the object. Thus, whether an object is visible not only depends on its distance from the observer, but also on its size, brightness, etc. Thus, a distance based organization of the objects would not be useful as it cannot be used to prune the search space.

Figure 2(a) shows a small window of an unbounded 2D universe. The dots represent objects and the line shows the path of the observer. Figure 2(b) shows all objects visible from position 1. Figure 2(c) shows all objects that are visible from some position on the path.



(a) All Data Points (b) Data Visible at Position 1 (c) Data Visible Along Path

Figure 2: All Data Points, Data Visible from Position 1, and Data Visible from Somewhere on the Path

During an immersive data exploration the observer will not see all points at once. With an unbounded universe it is even likely that the observer might not see most of the points, because a typical exploration will have to focus on some areas of interest while other areas remain unexplored.

2.1 Requirement Analysis

A navigating observer in an unbounded world imposes a number of requirements on the data to be extracted. We shape a set of requirements to the database system by analyzing different types of queries that the database system should support.

(Q1) objects that are visible Given an observer position the basic query is to extract all visible objects. With stringent real time requirements this type of query is not usually used during interactive data explorations. The query is a useful reference point, though. Among others it defines the objects that have to be retrieved.

Figure 2 illustrates the visible objects for a specific position (Figure 2(b)) and for all positions on a path (Figure 2(c)). Note that the observer is able to see distant objects, for example because they are large or bright. This does not encourage a distance-based organization of the access structure.

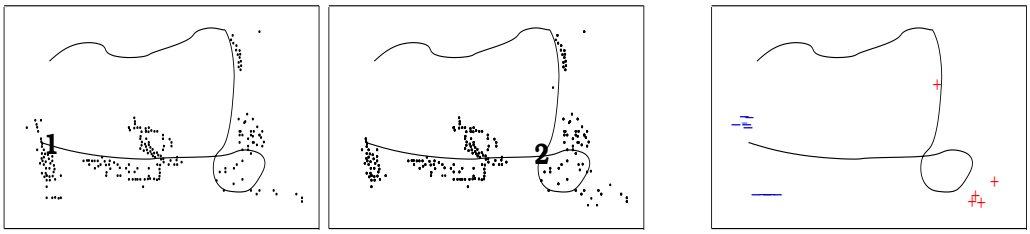
(Q2) objects of a specific visibility level Displaying all visible objects can quickly clutter the space and make it hard to analyze the data. For data explorations it can make sense to not show all visible objects or pursue a hierarchical approach where first only very visible objects are shown.

When mining data it could be relevant to not only extract objects with a visibility above a specific threshold but also objects with a visibility in a specific range to avoid visual domination of the most obvious relations.

Requirements Q1 and Q2 assume a single (static) observer position. They require an access structure that organizes objects according to their visibility. At the same time the access structure should be independent of the position of the observer to make it independent of the movement of the observer.

The next level are requirements imposed by a moving observer.

(Q3) objects that will become (in)visible When the observer moves the set of visible objects changes incrementally. Rather than recomputing the visible objects the database should offer a method to compute objects that become visible objects, Δ^+, and objects that become invisible, Δ^-.



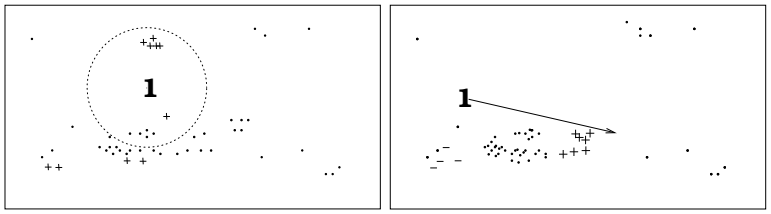
(a) Position 1 (b) Position 2 (c) Position 1 vs Position 2

Figure 3: The Observer Moves from Position 1 to Position 2

Figures 3(a) and 3(b) show the visible objects for two observer positions. Figure 3(c) shows the difference between the two positions. Clearly, the information exchange is with advantage restricted to objects that become visible (+) and objects that become invisible (-). Note the asymmetry between the two. While objects that become visible have to be passed on to have them visualized it is not mandatory to identify and communicate objects that become invisible.

(Q4) objects that are visible now or soon might be With an observer that moves continuously it makes sense to anticipate the movement and return visible objects as well as objects that soon might become visible when the observer moves. It is possible to distinguish a scenario with no information about the direction of the next move (cf. Figure 4(a)) and a scenario where we know the direction of the next move (cf. Figure 4(b)). If we do not know the

direction of the movement it is possible to return a superset of the actually visible objects. When the database system knows the direction of the movement it is possible to determine objects that will become visible and invisible, respectively.



(a) Unknown Movement (b) Known Movement

Figure 4: Anticipating The Movement of the Observer

This type of query is a generalization of query Q1. It fits a classical filter and refinement approach. The database system acts as a filter that returns a superset of the visible objects that is sufficiently accurate (and thus small) for the visualization system to work with.

(Q5) objects that at some point during the data exploration will be visible An important part of immersive data explorations are automatic fly-through scenarios where the system pre-computes a navigation path. Given such a path the database system can extract objects that at some point during the fly-through will become visible (cf. Figure 2(c)).

This list gives an impression of the types of observer relative queries that a database system should support to allow a tight coupling with a visualization system. The exact types of services that will be used will vary with the type of visualization. However, it is obvious that the extremes, passing on all the data and passing on the currently visible data, are not appropriate in the general case.

2.2 The Visibility Range

In order to retrieve visible objects it appears natural to associate each object with a visibility factor. The visibility factor is a metric for how well the observer can perceive an object. It depends on the properties of the object (size, position, color hue and brightness, etc) and the distance between object and observer. Let us assume that, given an object, O_i, and an observer, Obs, the visibility factor is proportional to the object size, $O_i.s$, and inversely proportional to the Euclidean distance, $|O_i.Pos - Obs.Pos|$, between object and observer:[1]

$$VF(O_i, Obs) = \frac{O_i.s}{|O_i.Pos - Obs.Pos|} \cdot c$$

The set of visible objects is the set of objects with a visibility factor above a certain threshold ρ:

$$visibleObjects(DB, Obs, \rho) = \{o \mid o \in DB \wedge VF(o, Obs) > \rho\}$$

Whenever the observer moves the visibility factor of all objects changes. Thus, the entire database or visibility-factor organized access structure has to be updated. We therefore associate

[1]The formula can be extended to incorporate additional object properties.

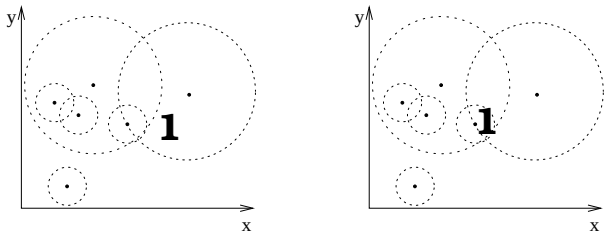
visibility ranges with each object. Visibility ranges are observer independent. This provides the basis for a static access structure (and allows for multiple observers).

Definition 2.1 (*Visibility Range (VR)*) Let O_i be an object. The visibility range associated with the object, $VR_i(O_i)$, is:

$$VR_i(O_i) = VF(O_i, Obs) \cdot |O_i.Pos - Obs.Pos| = O_i.s \cdot c \quad (1)$$

Thus, the visibility range is proportional to the object size (and possibly other object properties such as color hue and brightness) and independent of the observer position.

Figure 5 illustrates six objects with their associated visibility ranges. The objects are represented as black points. The dashed circles around the objects are visibility ranges. The observer position is marked with a cross.



(a) One Visible Object (b) Three Visible Objects

Figure 5: One Observer and Six Objects with Visibility Ranges

In Figure 5(a) the observer is within the visibility range of one object. Thus, only one object is visible. In Figure 5(b) the observer has moved to an area where the visibility ranges of several objects overlap. In this particular case three objects are visible.

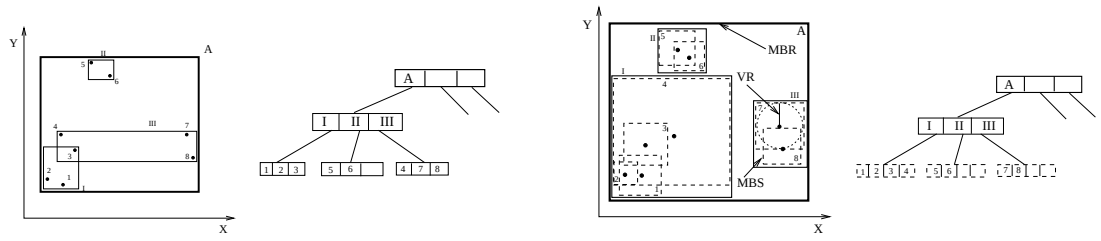
2.3 R- and VR-trees

To index spatial data the R-tree is used [10]. The R-tree indexes points, rectangles, lines and polygons by enclosing objects with Minimum Bounding Rectangles (MBRs).

Definition 2.2 (*MBR*) Let S be a set of rectangles. A rectangle r is a minimum bounding rectangle for S if all rectangles of S are inside r and no rectangle r' that is contained in r contains all rectangles of S.

The R-tree is a hierarchy of minimum bounding rectangles. The largest MBR, which contains all other MBRs, is placed at the root. Figure 6(a) shows a hierarchy of MBRs that binds 8 data points. The tree representation is shown in Figure 6(b). Each node consists of a set of MBRs and pointers to the MBRs at the lower level. The parent MBR contains all child MBRs. The leaf level consist of MBRs that enclose lines, points etc. and has pointers to the disk pages where the objects are located.

The *VR-tree* is an extended R-tree that indexes visibility ranges to support observer relative data extraction. To manage visibility ranges efficiently we add a node entry that uses squares to model position and visibility range of an object. The VR-tree creates a hierarchy of minimum



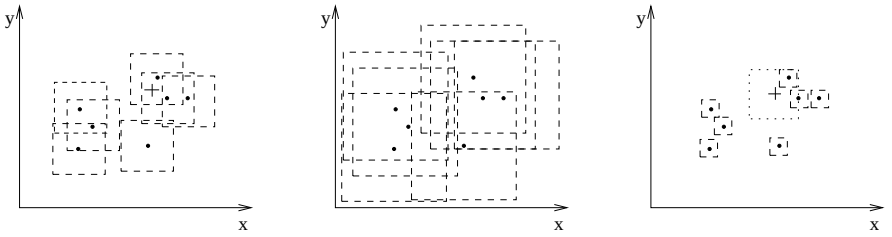
(a) MBRs (b) R-tree (c) MBRs + MBSs (d) VR-tree

Figure 6: Illustration of R- and VR-Tree

bounding rectangles (MBRs) like an R-tree. At the leafs minimum bounding squares (MBSs) are used instead of minimum bounding rectangles. Leaf nodes use MBSs to capture the spatial position and visibility range of objects. Internal nodes use MBRs to group several spatial objects as usual. Having two types of nodes increases the fanout. A leaf node can accommodate up to $\lceil \frac{b}{d \cdot p + s} \rceil$ entries. Here, b is the size of a disk page, $d \cdot p$ is the size of d-dimensional point, and s is a size of a minimum bounding square. An internal node can hold up to $\lceil \frac{b}{2 \cdot d \cdot p} \rceil$. For example, in the 2D (3D) case and with a page size of 8KB a leaf node may hold 341 (256) entries and an internal node 256 (170) entries. We assume 4 bytes for a value and a pointer. Figure 6(c) illustrates the visibility ranges graphically. At the leaf nodes each visibility range area is bounded by a minimal bounding square (MBS, dashed line). The edge of the square is equal to the visibility range diameter $2 \cdot VR$.

2.4 Algorithms

Visibility ranges decide whether an object is visible or not. We now show how visibility ranges can be scaled and queried to support different types of queries as discussed in Section 2.1. Figure 7 illustrates perfect, conservative, and optimistic visibility ranges. Perfect visibility ranges precisely delimit the area from where an object can be seen. Conservative visibility ranges overestimate the actual visibility area. As a result a query will typically also return objects that can not really be seen. Optimistic visibility ranges underestimate the actual visibility area. Thus, a query will not return all visible objects.



(a) $r = 1$ (b) $r > 1$ (c) $0 < r < 1$

Figure 7: (a) Perfect, (b) Conservative, and (c) Optimistic Visibility Ranges

A perfect visibility range does not scale the visibility range in Equation (1) ($r = 1$):

$$VR^p(O_i) = O_i.s \cdot c \cdot r, \quad r = 1$$

With perfect visibility ranges point queries that retrieve objects with MBSs that contain the query point return all visible objects. The following algorithm retrieves objects that are visible from point P. The function $mbs(c, s)$ returns a square with center point c and side length s, and $mbr(a, b)$ returns a rectangle defined by the points a and b.

Algorithm: VRtree-PointLookup

```
Input:
  Root of VR-Tree: N
  Observer position: Obs

Output:
  visible objects: vobj

FUNCTION P-lookup(N, P)
  FOR each entry e ∈ N DO
    IF isLeaf(N) AND (P ∈ mbs(e.C, e.s)) THEN vobj = vobj ∪ e
    IF ¬isLeaf(N) AND (P ∈ mbr(e.A, e.B)) THEN vobj = vobj ∪ P-lookup(child(e),P)
END FUNCTION

Body:
  vobj = ∅
  P-lookup(N, Obs)
```

A window query can be used if we want to retrieve objects that soon might be visible. This yields a buffer with respect to movement of the observer. Note though that the buffer is not precise as the direction of the navigation is not considered. Therefore, a window query returns objects that are almost visible. This will also include objects that actually become more invisible when the observer moves. Retrieving a subset of the visible objects is not directly possible with perfect visibility ranges.

Figure 7(b) shows conservative visibility ranges, which are defined as follows:

$$VR^c(O_i) = O_i.s \cdot c \cdot r, \quad r > 1$$

With conservative visibility ranges a point query retrieves too many objects. Window queries can be used to further increase the number of retrieved objects. This method is attractive if we want a robust filter in the sense that the set of visible objects shall remain valid if the observes moves within reasonable bounds. Conservative visibility ranges do not directly support the extraction of exactly the visible objects or the extraction of very visible objects. The algorithm for data extraction remains the same but more data is returned.

Figure 7 shows optimistic visibility ranges, i.e., underestimated visibility ranges:

$$VR^o(O_i) = O_i.s \cdot c \cdot r, \quad 0 < r < 1$$

With optimistic visibility ranges a point query returns only some of the visible objects—the very visible ones. Window queries can be used to increase the number of retrieved objects. With an appropriate query window size we are guaranteed to get all visible objects but invisible objects might also be included. Increasing the query window further will give a set of objects that remains valid if the observer moves within certain bounds.

Algorithm: VRtree-WindowLookup

Input:
  root of VR-tree: N
  observer position: Obs
  visibility range scale factor: r

Output:
  visible objects: $vobj$

FUNCTION W-lookup(N, W)
  FOR each entry $e \in N$ DO
    IF isLeaf(N) AND ($W \cap mbs(e.C, e.S)$) THEN $vobj = vobj \cup e$
    IF ¬isLeaf(N) AND ($W \cap mbr(e.A, e.B)$) THEN $vobj = vobj \cup$ W-lookup(child(e),W)
END FUNCTION

Body:
  $vobj = \emptyset$
  $ObsArea = mbs(Obs, max_{o \in DB}(o.s)/r)$
  W-lookup(N, $ObsArea$)

3 3D Probability Density Function Estimation

The probability density function (PDF) of a distribution in space is fundamental as a representation of this distribution. One of the most well-known and popular techniques for estimating such a PDF is the kernel density estimate. The kernel estimate [7] for a a set of observations, $(X_i, Y_i, Z_i), i = 1, \ldots, n$, at the point $\mathbf{x} = (x, y, z)$ is defined as follows:

$$\hat{f}_K(x, y, z) = \frac{1}{nh^3} \sum_{k=1}^{n} K\Big(\frac{x - X_i}{h}, \frac{y - Y_i}{h}, \frac{z - Z_i}{h}\Big), \tag{2}$$

where K is a kernel function. Traditionally it is assumed that $\int K = 1$, and $K(\mathbf{x}) = K(-\mathbf{x})$.

Various kernels K have been proposed in the statistical literature. Examples include square wave or Gaussian functions. It has been shown [31] that the accuracy of the estimation depends mostly on the smoothing parameter h and less on the choice of the kernel K. One can, for example, choose the value of h by minimizing the mean-square error:

$$\mathrm{MSE}\Big(\hat{f}_h(\mathbf{x})\Big) = E\Big(\hat{f}_h(\mathbf{x}) - f(\mathbf{x})\Big)^2,$$

where the expectation is taken with respect to the PDF f. Since the MSE is a function of $\mathbf{x}$ the optimal smoothing parameter is also a function of $\mathbf{x}$.

In order to minimize the MSE the best compromise between variance and bias has to be selected:

$$\mathrm{MSE}\Big(\hat{f}_h(\mathbf{x}) - f(\mathbf{x})\Big) = \mathrm{Bias}^2 \hat{f}_h(\mathbf{x}) + \mathrm{Var} \hat{f}_h(\mathbf{x}).$$

The multidimensional form of the Taylor theorem yields the approximations:

$$\mathrm{Bias} \hat{f}_h(\mathbf{x}) \approx 1/2 h^2 \nabla^2 f(\mathbf{x}) \int t_1 K(\mathbf{t})\, d\mathbf{t}$$

$$\mathrm{Var} \hat{f}_h(\mathbf{x}) \approx 1/n h^{-3} f(\mathbf{x}) \int K(\mathbf{t})\, d\mathbf{t}.$$

Hence, for a symmetric kernel K the optimal smoothing parameter is (cf. [28])

$$h_{opt}(\mathbf{x}) = \sqrt[7]{\frac{3f(\mathbf{x}) \int K^2(\mathbf{t})\ d\mathbf{t}}{\nabla^2 f(\mathbf{x}) \big(\int t_1 K(\mathbf{t})\ d\mathbf{t} \big)^2}}. \tag{3}$$

The optimal smoothing parameter as defined in Equation (3) clearly depends on the unknown PDF f. A number of heuristic approaches can be used to find the smoothing parameter. For instance, the optimal value for h can be computed for the normal distribution, and can then be used for the estimation of $\hat{f}_h(x)$ [28]. Other methods are a k-th nearest neighbor estimate [28], an adaptive varying window size [22, 14], and a bootstrap choice of the smoothing parameter [29].

3.1 The Kernel Computation Algorithm

We use a constant smoothing parameter, which minimizes the integrated mean-square error:

$$\text{MISE}\Big(\hat{f}_h(\mathbf{x})\Big) = E \int \Big(\hat{f}_h(\mathbf{x}) - f(\mathbf{x})\Big)^2\ d\mathbf{x}. \tag{4}$$

In this case the optimal smoothing parameter is

$$h_{opt}(\mathbf{x}) = \sqrt[7]{\frac{3 \int f(\mathbf{x})\ d\mathbf{x} \int K^2(\mathbf{t})\ d\mathbf{t}}{\int \nabla^2 f(\mathbf{x})\ d\mathbf{x} \big(\int t_1 K(\mathbf{t})\ d\mathbf{t} \big)^2}}. \tag{5}$$

The unknown underlying PDF in Equation (5) is approximated with a multivariate normal distribution. Since the accuracy of the estimation depends mostly on the smoothing parameter h and less on the choice of the kernel K, we are interested in a continuous kernel, which is inexpensive to compute. We use the Epanechnikov kernel (cf. Equation (6)) since it requires a small number of arithmetic operations in comparison to, e.g., the Gaussian function, which requires an exponential function approximation.

$$K(t_1, t_2, t_3) = \begin{cases} 15/(8\pi)(1 - (t_1^2 + t_2^2 + t_3^2)) & \text{if } t_1^2 + t_2^2 + t_3^2 \leq 1 \\ 0 & \text{else} \end{cases}. \tag{6}$$

To calculate the PDF values on the grid points we have to scan the database or an appropriate sample twice. The first scan is used to calculate the estimation parameters (variances and h_{opt}). For this step it makes sense to exploit existing database functionality. For example with Oracle the SQL query

```
select variance(x), variance(y), variance(z), count(*)
from r sample block (5)
```

computes the variances and n for a sample that includes 5% of the database blocks. With a well designed and maintained database this is considerably faster than a full database scan. The second scan is used to compute the actual PDF estimation. Since the kernel function in Equation (6) is equal to 0 in the area $t_1^2 + t_2^2 + t_3^2 > 1$ only observations that fall into the area $\{(t_1, t_2, t_3) : (t_1 - x)^2 + (t_2 - y)^2 + (t_3 - z)^2 \leq h^2\}$ influence the estimated PDF value at point (x, y, z).

The PDF is implemented as a three dimensional data cube with dimensions x, y, and z. The cardinality of each dimension is g. The individual steps of the kernel computation are given in the following algorithm.

Algorithm: PDF_Estimation

Input:
  Database with n observations: $(X[i], Y[i], Z[i]),\ \ i = 1, \ldots, n$
  Number of grid points in each dimension: g

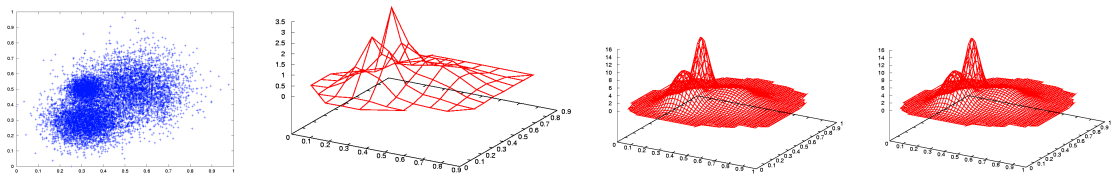
Output:
  Data cube with PDF values on grid points: $PDF[i, j, k]$

Body:
  Initialize PDF
  Calculate h_{opt} according to Equation (5)
  FOR $i = 1$ TO n DO
    $\mathcal{A}_g$ = set of PDF points that are influenced by data point $(X[i], Y[i], Z[i])$
    FOR EACH $(k, l, m) \in \mathcal{A}_g$ DO
      $PDF[k, l, m] = PDF[k, l, m] + K\left(\frac{k - X_i}{h}, \frac{l - Y_i}{h}, \frac{m - Z_i}{h}\right)$

The cost of the algorithm is $7 * h_{opt} * g^3 * n$ where n denotes the number of observations (database size), g denotes the number of grid lines along each dimension, and 7 derives from the number arithmetic operations in Equation (6) (three multiplications, three additions, and one subtraction). The size of the PDF data cube is $4 * g^3$ (size of PDF value, three dimensions).

3.2 Evaluation of the PDF Estimation

Since it is not possible to plot a four dimensional PDF we use a two-dimensional dataset to illustrate the PDF estimations for different numbers of grid lines (cf. Figures 8). As g increases we get a more precise PDF estimation. The figures show that a number of grid lines around 30 is reasonable.

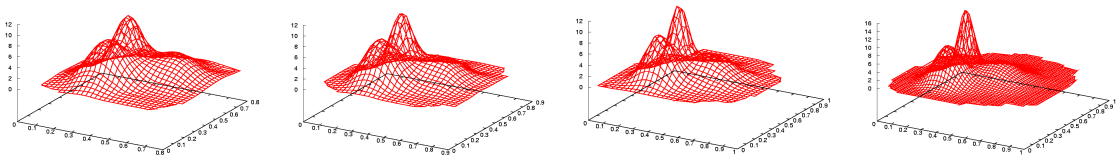


(a) Dataset (b) $g = 10$ (c) $g = 30$ (d) $g = 50$

Figure 8: 2D PDF for different g

Figures 9 shows the impact of data set size n to PDF estimation. As n decreases the impact of any point in data set to PDF increases. This yields flatter picture for small n. The figures shows, that n between 5'000 and 10'000 is reasonable.

In order to evaluate the estimated PDF numerically one can use the MISE error metric (cf. Equation (4)), or the estimated MISE when the underlying probability density function is unknown. The (E)MISE is a common error metric, because it is easy to investigate mathematically (e.g., for solving minimization problems). Its disadvantage is the interpretation of the numeric results. We define an alternative error metric, which is based on the average error at any grid



(a) $n = 1'000$ (b) $n = 2'000$ (c) $n = 5'000$ (d) $n = 10'000$

Figure 9: 2D PDF for Different Sample Sizes n

point:

$$AE_{PDF} = \frac{1}{g^3 \max_{x,y,z} \hat{f}_{h,g}(x,y,z)} \sum_{i,j,k=1}^{g} |\hat{f}_{h,g}(i,j,k) - f_(i,j,k)|. \quad (7)$$

We approximate the unknown density function f with its estimation $f_{h_{opt},\bar{g}}$ and a high value of $\bar{g}$. Table 1 shows the empirical results for AE_{PDF}. For the computation we assumed $h = h_{opt}$, $n = 10000$, and $\bar{g} = 100$.

Estimation	$g = 50$	$g = 30$	$g = 20$
AE_{PDF}	0.01001	0.0176	0.0301

Table 1: Numeric Evaluation of AE_{PDF} Error Metric

Note that even for a small number of grid lines the error is low. For example, for $g = 30$, the error is below 1%.

4 Space and Objects—Perceptual and Cognitive Considerations

With respect to the visualization of information one of our main aims is to create a visual space and partly fill in with an array of special objects created for the purpose. These objects have features, which denote values of selected parameters in the database under scrutiny. From a perceptual point of view we have to deal with the space and its objects as separate entities in their own right and as referred to in language. In the following we present a short overview of this structure.

Whether in real reality or in virtual reality with its objects two basic questions immediately arise: “Where” do we see something of interest and “what” is it? Neuropsychological imaging, the study of brain cells, and the study of patients with special brain damages have shown that the where-stream and the what-stream are separate processes. Besides, it is an old philosophical tenet that space (and time) are given a priori or concurrent in order for a living being to be able to process what are given to the senses.

4.1 Axes and Frames of Reference

In our project it is possible to use different visualization devices. In the “cave” which is a cube with stereoscopic projections on all six sides one of the persons can be tracked as to the position of the head (only few persons can be accommodated and they should preferably follow the tracked person). This immersive virtual reality is complemented by a 3D panorama theater (curved screen)

and a 3D “power wall” (flat screen). But work done via the ordinary computer screen have also been very useful. It allows a quasi-3D interactive presentation with a moving space.

As a common feature for the visualization devices the space can be set into motion. By zooming in or out the observer sometimes gets the impression of being the active part traveling into or in the virtual space (although hard to achieve before the computer screen). Being immersed in a more or less continually moving world it is difficult and soon impossible to keep track of the positions visited and the route followed. This raises the question of axes or frames of reference and their accessibility.

Jackendoff [12] proposed to distinguish between two sorts of frames: intrinsic and environmental. Intrinsic frames belong to the objects in the space while environmental frames belong to the space itself or the observer. Objects have certain geometric properties, such as axes or parts, which might help in recognition and establishing their orientation. Other axes may be imposed by well-known objects, for instance by houses or cars that exhibit a “canonical” orientation and parts with special significance (among other things doors and windows as canonical encounters). But information visualization usually operates with simple geometric forms with no obvious canonical properties. They are easily recognized and assuming that they are stationary they might show if the space is moving (rotating, zooming) by means of their axes and relative sizes. Apart from this they contribute little or nothing to the impression of up and down in their environment. Another intrinsic frame is the movement of the object that establishes some relation to other objects in the given space. If this kind of movement is to be used it should be specific for local groups of objects to show some underlying property (we return to this).

Environmental frames interact with the frames established by the objects. Gravitation and the canonical orientation of a house are normally in line. An observer standing at the house imposes still a new frame as a point of view for a description of the scene; there is a front and a relative estimation of size. As is the case for gravitation the geographical frame of reference grows out of physical considerations. In the scene just mentioned there is normally a fixed and regular relationship among the frames. Sitting in an airplane circling around this is not always the case. Gravity may pull the passenger in a direction not in conformity with the geographical frame (the horizon) below the airplane. Well-known objects (houses etc.) with their canonical orientation follow the horizon in accordance with the contextual frame of reference. The passenger follows another context, i.e. canonical relationships and gravity inside the airplane. This separation of sets of frames does not create problems; it seems to be an interesting feature at least for many people.

In the virtual space constructed by data miners the situation is more difficult. Placed for instance in the panorama with the artificial objects all around in the visible field the observer clearly senses the gravitational frame and generate (maybe implicit) the geographical frame. This is a purely local occurrence with no clues in the virtual space as to possible frames of reference. When the space is brought into motion it is not like flying in an ordered world. Objects with their geometrical frames whirl around and seem to follow a contextual frame (unless they are moved independently). If their axial orientation is fixed and a sufficient number still in sight some sense of the imposed up and down is preserved. Even if some kind of a coordinate system may be (partly) visible as an envelope for the space an important next step is the creation of visible and stable indications of frames of reference belonging to the space itself. As the space has no time of its own it is difficult to judge the relative distances of objects when the space moves. An (arbitrary) indication of the fixed speed and a time connected with the space should make it easier to travel around and estimate distances between objects or the clusters around (virtual stopwatches might be useful).

4.2 Objects and their Properties

According to the well-established Gestalt laws objects are readily grouped on account of the their appearance and the relations holding between them. This series of dots is an example: ... ... It is nearly impossible not to experience them as grouped into clusters of three dots. Similarity is like the proximity a strong organizing factor. “Common fate” is the kind of similarity induced by movement. If a selection of objects in the visible field follow parallel trajectories they are prone to be perceived as belonging to the same surface or be parts of a common object; in this case we also see another Gestalt law in action: the whole is more than its parts (even fish in a shoal may be perceived as forming an object with new, i.e., emergent properties).

If all of the objects in the virtual data mining space (panorama or cave) is set into uniform motion they might form a shoal but the emergent properties belong to the full extension of the assembly and we only learn something about the space. It is typically experienced only as long as it is filled with objects (with an intuitive sense of a void beyond). This is by and large only informative if we want to visit different locations or by the movement is given some kind of axes in this space, as mentioned above. But selected objects could be made to form a shoal and thereby exhibiting some parameter range hidden in the database. Position in time and space are essential features as stated by Scholl [25, p. 24]: “As a whole, the evidence relating objects and features suggests that object-based processing may often trump feature-based processing, but that not all features are created equal: in some circumstances, spatiotemporal features may be more tightly coupled with object representations than are surface-based features such as color and shape.”

He is referring to multiple objects moving around and sometimes getting out of sight. When the attentional load is high other features are largely sorted out. In textbooks on information visualization it is common to meet recommendations regarding a multitude of possible object features such as color, luminance, shape and orientation. In addition a warning is regularly sounded: they should be deployed with great caution. Conjunctions of many features make it difficult to group the objects and search time augments in a steep curve. A moderate number of surface features are to be recommended after deciding on the layout of spatiotemporal relations. Assigning only two values to shape, size, color, texture, and orientation amounts to thirty-two kinds of objects to be distinguished and possibly grouped in the stationary or moving space.

Therefore, we operate with a much smaller number, as for instance a tetrahedron in two sizes and two colors distributed along the three spatial coordinates. It should be noted that size and luminance are features to be perceived as constant in a 3D space, which means that the computation behind the visualization includes the distance and angle from the observer. Similar dependencies apply in an important way to texture as an important feature in the formation of objects (edges, curvature) and their distance and orientation. In theories of perception one of the main currents is based on what is called “direct perception” of the information in the optic array and its change when the observer moves along. This is experienced with the collection of the objects in their wholeness as an optic flow with its emergent axes connected with the movement. This brings a new frame of reference into play like the one behind the experiences in the airplane as passenger, mentioned above. In order to accomplish this and get the corresponding overview some sort of horizon should be established.

Surfaces are important in the formation of objects and they presumably belong to early stages in the visual processing. Grouping operates in the way described by the Gestalt laws and the theory of direct perception toward the creation of putative objects in the field. As part of the process a ground is established, at times resulting in a curious ambiguous figure-ground segregation (a well-

known example is the vase/face figure shifting between two opposing faces and a vase). Below the level of conscious percepts grouping seemingly also operates on objects. Scholl [25, p. 19] relates complex objects and multi-object units in this way: "Visual surfaces constitute another level of representation which can encompass both of these categories: complex objects can consist of multiple surfaces, while multiple objects can be arrayed along a single surface." Several visual phenomena are based on "the perceived interpretation of the visual field in terms of surfaces." (p. 19). We use this tendency to form surfaces out of collections of objects and operate along them by creating semi-transparent envelopes around regions above some level of density (as described in Section 5 and [19]). Up till now it has been very successful on the computer screen but it might as well help in grouping objects in the immersive virtual reality space.

4.3 Where and What?

When observing the tetrahedrons distributed in the space around us in the panorama a coworker observed something like a spiral. This is what visual data mining is all about: detection of unsuspected patterns and their description. But the language has its limits. "Don't you see it over there to the right?" Well, maybe we can establish a common frame of reference (axes) and still the spiral may elude our perceptual capabilities. Then we set the space into motion and travel toward the phenomenon but this zooming in is also a loss of perspective. "It is on the cluster below!" Prepositions such as "on" or "in" (locatives) are rather undetermined in a purely linguistic context. Nonetheless, they might lead the way if supplemented by a visual input. "On" denotes a geometric relationship but often functional properties have to be taken into consideration. Being on something means being to a certain extent controlled by this thing (e.g., the picture is on the wall). One way to decide if the spiral were on something else would be to mark the elements of this something (as can be done on a computer screen), combine them to an object, and make the computer program move such an object. The spiral could by a pointing device be marked in another way and should follow the "something" if it is controlled and thereby correlated with it.

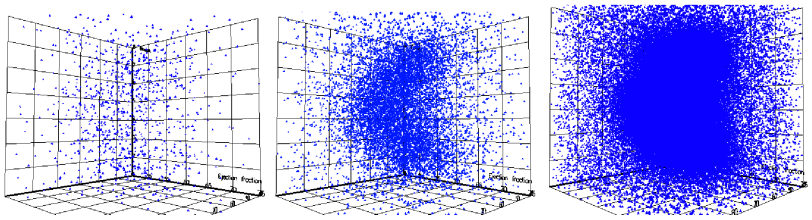
Even if many interesting structures has been discovered in the databases at our disposal there are a lot of opportunities for developing the search capabilities further. Important areas are on the one hand orientation when navigating in the space and on the other delimiting objects (clusters) and describing their relationships.

5 Nested Surfaces

Employing scatter plots to view data is intuitive as each observation is faithfully displayed. However, scatter plots hit limitations if the dataset is big, noisy, or if it contains multiple structures (cf. Figure 10). With lots of data the amount of displayed objects makes it difficult to detect any structure at all. Noise easily blurs the picture and can make it impossible to detect interesting relationships. With multiple structures it often happens that one structure is more pronounced than another. In this situation the less pronounced structures easily gets lost. For the purpose of data mining this is particularly bad as it is usually the less obvious relationships that are the most interesting ones.

As an example consider the Spiral-Line data set presented in Figure 10. The data set consist of a vertical line in the middle (40% of all observations), a spiral curve around the line (40% of all observations) and uniformly distributed noise (20% of all observations). The data points around

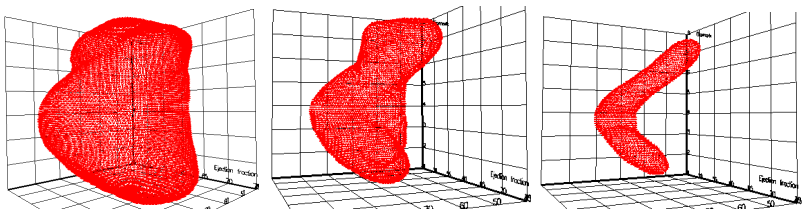
the spiral curve form the most dense and notable region. Since the data points around the vertical line have a higher spreadness it is easy to treat it is a noise and not pay attention to it.



(a) $n = 1'000$ (b) $n = 10'000$ (c) $n = 100'000$

Figure 10: Spiral–Line Data Set for Different Databse Sizes

The direct display of 3D data points does not support the detection of these structures. To enhance the scatterplot visualization we replace (or complement) the data points with points that form density level surfaces. Figures 11(a) to 11(c) present the surfaces for different density levels α. Figure 10). Figure 11(a) shows the surface for the lowest density level. This Figure can be used for the detection of outliers. Figures 11(b) and 11(c) show surfaces for higher density levels. Together with Figure 11(a) they emphasize the structure of the data set. The surface in Figure 11(b) clearly identifies the vertical line and the spiral (the quality is much better on the monitor).



(a) $\alpha = 1/10m$ (b) $\alpha = 5/10m$ (c) $\alpha = 9/10m$

Figure 11: Spiral–Line Data Set and Associated Surfaces. m is the Maximum Density in the Data Set

5.1 Definitions

We use a topological approach to define a surface. A surface is a set of points if and only if the neighbourhood of any point is similar to a two–dimensional open disk. More formally, we define:

Definition 5.1 *(Elementary surface)* Let f be a function that maps an open disc D^2 to a set of points C. C is an *elementary surface* if and only if f is homeomorphic.

Definition 5.2 *(Surface)* A *surface* is a connected set of points if and only if the neighbourhood of any point of the surface is an elementary surface.

A border is a set of points: $\partial C = [C] \backslash C^\circ$ where $[C]$ contains the limit points of C and C° contains the inner points of C. To show that ∂C is a surface one needs to find a parametrisation function that maps an open disk D^2 into ∂C. The existence of a parametrisation function in our case follows from the implicit function theorem if $\nabla f(x, y, z) \neq 0$ for all $(x, y, z) \in \partial C$.

5.2 Algorithms

This section gives two algorithms to compute nested surfaces: `Surface_GridPoints` and `Surface_GridLines`. The `Surface_GridPoints` algorithm calculates the border $B = \partial\{(x, y, z) : f(x, y, z) \geq \alpha\}$. The basic idea of the algorithm is to scan the PDF and compare each value against its neighbors: if the value is greater than α and there exists a point in the neighborhood that is less than α then the value is added to B.

Algorithm: Surface_GridPoints

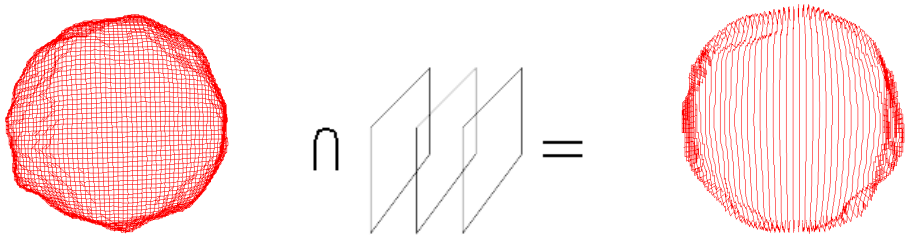
Input:
  Number of grid lines per dimension: g
  Data cube with PDF grid point values: PDF
  Density level: α

Output:
  Surface grid: B

FUNCTION IsBorderPoint(PDF,i,j,k)
  RETURN ($PDF[i,j,k] \geq \alpha$) AND ($PDF[i',j',k'] < \alpha$)
    for some $(i',j',k') \in (i+h_1, j+h_2, k+h_3)$ where $h_1, h_2, h_3 = -1, 0, 1,\ \ |h_1|+|h_2|+|h_3| = 1$
END FUNCTION

Body:
  $B = \emptyset$
  FOR $i, j, k = 1$ TO g DO
    IF IsBorderPoint(PDF, i, j, k) THEN $B = B \cup PDF[i,j,k]$

The `Surface_GridLines` algorithm extends the `Surface_GridPoints` algorithm. The main idea of the algorithm is to draw contour curves on the surface. These curves, in turn, are calculated by intersecting a surface with cutting planes parallel to the XY, ZY, and ZX planes. The idea of the plane curve calculation is presented in Figure 12. We scan the PDF values with a condition $i = i_0$ for ZY planes, $j = j_0$ for ZX planes, and $k = k_0$ for XY planes.



(a) Surface (b) Intersecting Planes (c) Vertical Grid-Line

Figure 12: Grid-Line Surface

The details of the computation of an individual grid line are illustrated in Figure 13. Figure 13(a) shows a cutting plane. Border points are shown as filled circles, inner cluster points as plus signs, and outer cluster points are not shown. The algorithm connects the border points to form a polygon curve. For each PDF border point we are looking for PDF border points in the directions presented in Figure 13(b). Note, that we scan PDF from left to right and from bottom to top. Therefore, there is no need to draw lines to the bottom and to the left. We draw vertical and

horizontal connections between border points. For diagonals we make additional checks. We do not draw a diagonal line if there are two lines in its neighborhood (cf. Figure 13(c)). With this we avoid squares with crossing diagonals.

(a) Connecting Border Points (b) Valid Directions (c) Final Grid Line

Figure 13: Grid-Line Computation in an Intersecting Plane

The individual steps of the ZY plain curve calculation are presented in the `ZY_Plane_Curve` algorithm.

Algorithm: ZY_Plane_Curve

```
Input:
  ZY plane number: i_0
  Data cube with PDF grid point values: PDF
Output:
  polygonal contour line on ZY plane at level i_0: C = C^ZY_{i_0}
Body:
  C = ∅,  i = i_0
  FOR j,k = 1 TO g DO
    IF IsBorderPoint(PDF,i,j,k) THEN
       IF IsBorderPoint(PDF,i,j+1,k) THEN C = C ∪ line(i,j,k,  i,j+1,k)
       IF IsBorderPoint(PDF,i,j,k+1) THEN C = C ∪ line(i,j,k,  i,j,k+1)
       IF IsBorderPoint(PDF,i,j-1,k+1) AND ¬IsBorderPoint(PDF,i,j-1,k) AND
          ¬IsBorderPoint(PDF,i,j,k+1) THEN C = C ∪ line(i,j,k,  i,j-1,k+1)
       IF IsBorderPoint(PDF,i,j+1,k+1) AND ¬IsBorderPoint(PDF,i,j+1,k) AND
          ¬IsBorderPoint(PDF,i,j,k+1) THEN C = C ∪ line(i,j,k,  i,j-1,k+1)
```

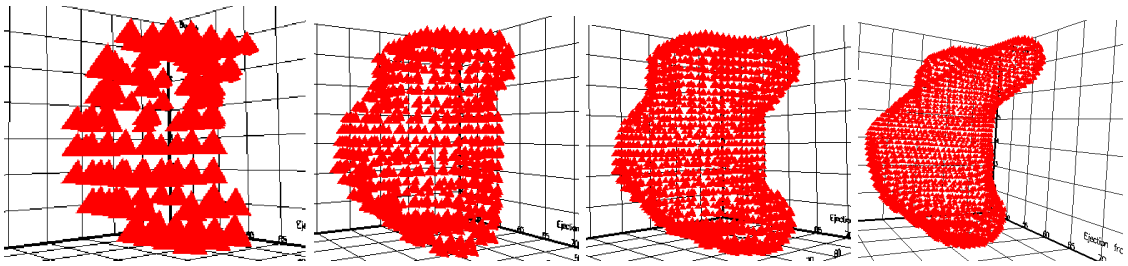
Algorithm: Surface_GridLines

```
Input:
  Data cube with PDF grid point values: PDF
Output:
  Contour lines on the surface: C
Body:
  C = ∅
  FOR i = 1 TO g DO C = C ∪ ZY_PlaneCurve(PDF,i)
  FOR j = 1 TO g DO C = C ∪ ZX_PlaneCurve(PDF,j)
  FOR k = 1 TO g DO C = C ∪ XY_PlaneCurve(PDF,k)
```

We use the `Surface_GridPoints` method to illustrate surfaces on 2D devices while we use the `Surface_GridLines` method for surfaces in immersive 3D environments.

5.3 Evaluation

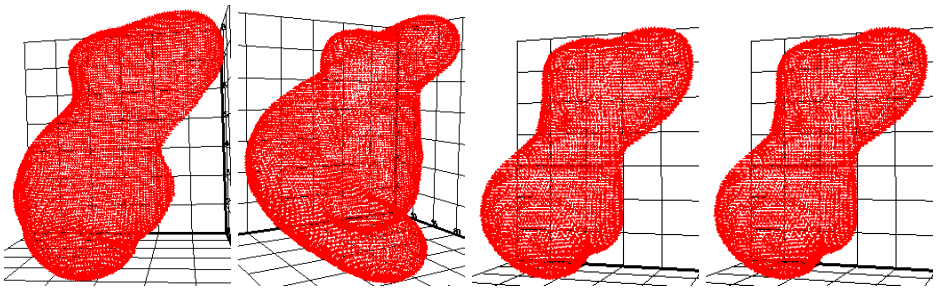
We use the three-dimensional scatter plot in Figure 10 and a single level surface for different number of grid lines (Figures 14(a) to 14(d)) to evaluate surfaces. In order to get a fair visual comparison of the influence of g on the quality of the surface the size of tetrahedra depends on g. It is chosen so that the tetrahedras are always near each other visually. It is easy to see (Figures 14(a) and 14(b)) that $g = 10$ and $g = 20$ are not enough for a nice surface—there are two few tetrahedras at the ends of spiral curve. As g reaches 30 the picture becomes detailed enough.



(a) $g = 10$ (b) $g = 20$ (c) $g = 30$ (d) $g = 50$

Figure 14: Cluster Surface for $\alpha = 1/10m$ for Varying Values of g

Figure 15 presents the impact of the size of the data set on the surface quality. The figures show that $n = 10'000$ is sufficient for a nice surface. Figure 15(b) is drawn from a different perspective that shows the unevenness of the vertical line for $n = 5000$. Note, that in contrast to the scatterplot nested surfaces are not overloaded as n increases (compare Figure 10(c) with Figure 15(d)). The quality improves monotonically as the number of observations increases.



(a) $n = 1'000$ (b) $n = 5'000$ (c) $n = 10'000$ (d) $n = 100'000$

Figure 15: Cluster Surface for $\alpha = 1/10m$ and Varying Values of n

We use an approach similar to Section 3.2 to evaluate nested surfaces numerically. Equation (8) defines the quality measure of the surfaces. It shows the average error we make at any point (i, j). s is the parameterization function that maps the open unit disk D^2 to ∂C_α. Since s is usually unknown we approximate it with $\hat{s}_{\bar{g}}$ with large value of $\bar{g}$.

$$AE_S = \frac{1}{g^2 \max_{x,y,z} \hat{f}_g(x,y,z)} \sum_{i,j=1}^{g} |\hat{s}_g(i,j) - s(i,j)|, \tag{8}$$

Table 2 presents the numbers for AE_S with $\bar{g} = 100$. The AE_S error shows that the error is below 1% if the number of grid lines is above 30.

1/10m	0.0289	0.0083	0.0045
5/10m	0.0249	0.0071	0.0038
9/10m	0.0069	0.0011	0.0005

Table 2: The AE_S Error for Different Number of Grid Lines

6 Empirical Evaluation

Table 3 shows the numerical break down of the 3DVDM system in Figure 1. Our main interest is the relative comparison of the different parts. The absolute values are less relevant and vary with the chosen architecture. The experiments were calculated on the Pentium III 1GHz PC computer with 512MB of main memory.

number of objects in the database	time (sec) to extract visible objects	*number of visible objects*	time (sec) for PDF estimation g=30(20,50)	*size (KB) of estimated PDF g=30(20,50)*	time for 4 nested density surfaces	*size (KB) of 4 nested density surfaces* g=30(20,50)
250.000	1	*1.000*	2 (<1, 10)	*108 (32,500)*	<1	*73 (30, 219)*
500.000	2	*2.000*	3 (<1, 14)	*108 (32,500)*	<1	*73 (30, 219)*
1.000.000	4	*3.000*	4 (<1, 18)	*108 (32,500)*	<1	*73 (30, 219)*
1.000.000	5	*5.000*	6 (2, 25)	*108 (32,500)*	<1	*73 (30, 219)*
1.000.000	11	*10.000*	9 (3, 38)	*108 (32,500)*	<1	*73 (30, 219)*
1.000.000	25	*100.000*	40 (15, 149)	*108 (32,500)*	<1	*73 (30, 219)*

Table 3: Numerical Evaluation of the 3DVDM System

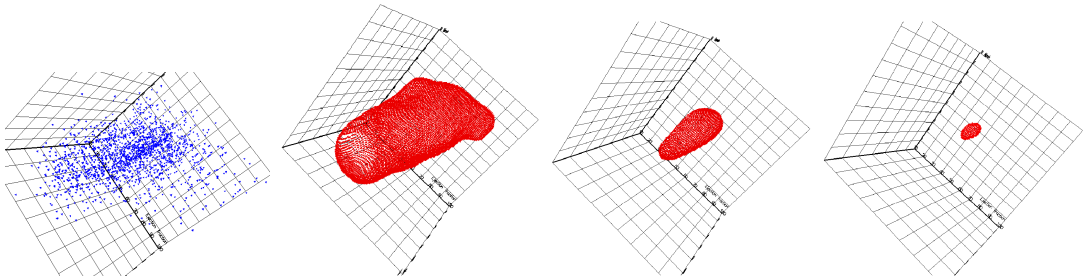
The evaluation shows that the database part is robust with respect to the size of the data set. Thus, the indexing of visibility ranges with the VR-tree is effective. The cost of the database part mostly depends on the number of objects that have to be extracted. The PDF estimation is expensive, and it is important to reduce the size of the data early. Whenever feasible the PDF estimation should be computed for a small sample or the visible objects only. The time of the PDF estimation also depends on the variance of the data set. With a large variance the displayed times increase up to a factor 4. It is also important to keep the number of grid lines low. The number of grid lines affects the computation time and the size of the PDF estimation. In contrast, the size of the PDF estimate is independent of the size of the database. Because the size of the density estimate is quite small when compared to the size of the database it is not usually a problem to store precomputed PDF estimates in the database to speed up data explorations. The computation of density surfaces from density estimates is very fast, and as it is not significantly cheaper to store surface information instead of density estimates, the surfaces should be computed on the fly.

The second part of this section visually illustrates nested surfaces for a real and an artificial data set. For each data set we show nested surface grids and offer an interpretation. Note that the visual information in the printed images is limited as three dimensions have to be projected into

two. Also nested surfaces have to be shown in figures side-by-side. The reader may download and install the 3DVDM system to experiment with the surfaces.

The heart dataset is used to investigate the mortality of humans. The dataset consists of 1755 observations. For each person the age, the ejection fraction, and a biomarker are recorded. The ejection fraction of the heart is commonly used to predict the mortality. Measuring the ejection fraction is expensive, though. The biomarker is extracted from a comparably cheap blood sample. The question then is whether and how ejection fraction and biomarker relate.

Figure 16(a) shows a 3D scatterplot of the heart dataset. It is difficult to extract any information from this plot. The situation improves if we rotate the plot. However the spreadness of the data still makes it quite difficult to interpret the data.



(a) Data (b) $\alpha = 1/10m$ (c) $\alpha = 5/10m$ (d) $\alpha = 9/10m$

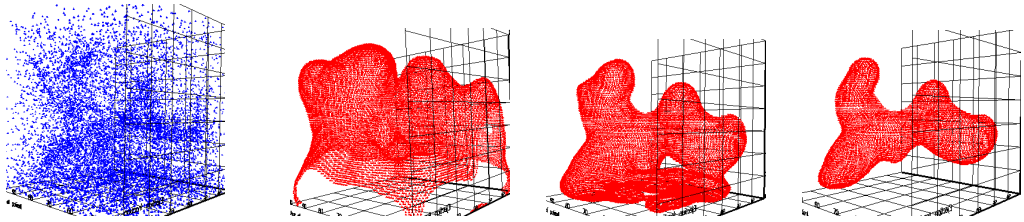
Figure 16: The Heart Dataset

Figure 16(b) shows the surface for $\alpha = 1/10$ of the maximal PDF value. The surface encloses all data points except outliers. Since there are only a few data points near the border one can easily see the non-evenness of the surface. While the printed version of the surface is still difficult to interpret an animated or immersive visualization shows that the surface resembles a spiral in the three-dimensional space. Many people are not able to see the spiral when looking at the raw data—even when told that the data is scattered along a spiral. When looking at the surface the spiral is much easier to detect. Figures 16(c) and 16(d) shows surfaces that enclose more dense regions of the heart dataset. Note that on the screen these surfaces are plotted inside each other. The nested plotting emphasizes the structure of the dataset. For higher densities the data should be approximated by three dimensional line segments rather than a spiral.

Summarizing, it is likely that there is a relation between age, ejection fraction and biomarker. The surfaces suggest a combination of a linear model (for $\alpha \geq 5/10m$) and a spiral (for $\alpha < 5/10m$).

The artificial data set contains three structures: 1) points, which are spread around a randomly generated polygonal line, 2) a 3D structure defined in terms of a simulated random variable: (uniform(0,1), uniform(0,1)), and 3) uniform noise.

Obviously, it is quite difficult to understand the structure from the scatter plot in Figure 17(a). The nested surfaces in Figures 17(b) to 17(d) emphasize and clarify the structure.



(a) Dataset (b) $\alpha = 1/10m$ (c) $\alpha = 3/10m$ (d) $\alpha = 5/10m$

Figure 17: Artificial Dataset

7 Related Work

Although data mining is inherently interdisciplinary most data mining systems employ coarse interaction patterns [4, 16, 5, 24]. The main reason is that the systems often evolve from specialized systems targeted at one area only. We pursue an approach that tightly integrates mature components from different areas and aims at a fine-grained interaction.

Database index structures are based on the R-tree [10]. The X-tree is an extension of the R-tree to index high dimensional data [2]. Other R-tree variants include the SS-tree, which is used for similarity searches in multidimensional data [34], or the SR-Tree, which supports nearest neighbor searches [13]. Another family of access structures partitions the space. The kdB-tree is the classical representative and is an access structure for high dimensional data [21]. The LSD^h-tree uses a two level directory structure for similarity search in feature vectors [11]. Common to all these approaches is a spatial grouping of objects. In contrast the visible objects considered in this paper are not usually co-located in space.

Little work has been done on dynamic access structures. Šaltenis at al. [23] introduced the TPR-tree to index the position of continuously moving objects. All objects move independently and have an associated speed and direction vector. We assume in some sense the opposite: a static world with a moving observer and rigid real time requirements.

Probability density functions and kernel estimation have been used for several years in mathematical statistics [26, 28, 31, 8, 6, 29, 22, 14]. Despite the solid mathematical underpinning of the kernel estimation method it is rarely used in database contexts (a notable exception is [3]). Database systems typically use histograms [1, 17] to, e.g., estimate the selectivity of queries. The kernel method generalizes histograms methods and guarantees a probability density function with a continuous derivative. This is necessary for the definition and computation of surfaces. We focus on a numerical and visual evaluation of the kernel method that provides guidelines for the selection of an appropriate database sample size and number of grid lines. The results are worked out for 3D data and the optimal smoothing parameter.

Surfaces have received some attention in the visualization community [27, 35, 18] where the main focus are methods and data structures that support the effective and efficient visualization, e.g., fast rendering, transparency, and lightening. In [33] various aspects of density surfaces have been studied, including lightening, transparency stereoscopy, dynamic rotation, and dynamic visualization techniques. We are interested in the defining properties of surfaces that are easy to perceive. The human perceptual system is apt at processing large quantities of surface information [30] but tends to have difficulties recognizing sets of individual data points that define these surfaces. Thus, surfaces are a natural part of 3D environments, whether immersive (real or virtual)

or animated on the screen, as they are picked up well by the human perceptual system.

8 Summary and Future Work

This paper discusses an interdisciplinary approach to build a data mining system, with fine-grained interaction patterns between different disciplines. First, the paper surveys the visibility range of objects that is used to improve existing database access structures. We then use insights from cognitive psychology and kernel estimation to enhance scatterplots with objects that emphasize the structures in the data. Nested surfaces can be used to detect more and less pronounced structures within a single data cluster. Throughout, we provide detailed algorithms and empirical evaluations of our methods, both visually and numerically.

The initial results of our approach are very promising. In the future we have planned to consolidate the observer relative data extraction, design density-based fly-through scenarios, and further exploit the advantages of (animated) VR. At the technical level we want to improve the interaction between the different parts of the system to support real time immersive data explorations for large databases. To achieve this we want to minimize the amount of information that has to be communicated and progress the individual parts of the system. For example, we want to implement an adaptive kernel estimation that minimizes the number of grid lines. At the conceptual level we want to progress the understanding of space and objects. Our ultimate goal is to detect hidden structures during immersive data explorations. Further case studies are needed to get a robust understanding of how scale, orientation in space, and object properties influence the recognition of structures. We also want to develop algorithmic solutions for visually successful data analyses. An example is an algorithmic solution that separates the more and less pronounced structures within a single data cluster.

9 Acknowledgments

The work was supported in part by the Danish Research Councils through grant 5051-01-004. We are grateful to Jan Parner for the heart data.

References

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: building histograms without looking at data. In *Proceedings of the ACM SIGMOD 1999 International Conference on Management of Data, June, Philadephia, Pennsylvania*, pages 181–192, 1999.

[2] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree : An Index Structure for High-Dimensional Data. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 28–39. Morgan Kaufmann, 1996.

[3] B. Blohsfeld, D. Korus, and B. Seeger. A Comparison of Selectivity Estimators for Range Queries on Metric Attributes. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *Proceedings ACM SIGMOD 1999 International Conference on Management of Data, June, Philadephia, Pennsylvania, USA*, pages 239–250. ACM Press, 1999.

[4] A. Buja, D. F. Swayne, and D. Cook. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, Vol. 5(No. 1):78–99, 1996.

[5] M. P. Consens and A. O. Mendelzon. Hy+: A Hygraph-based Query and Visualization System. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 511–516. ACM Press, 1993.

[6] G. C. Van den Eijkel, J. C. A. Van der Lubbe, and E. Backer. A Modulated Parzen-Windows Approach for Probability Density Estimation. In *IDA*, 1997.

[7] L. Devroy and L. Gyorfi. *Nonparametric Density Estimation*. Jon Wiley & Sons, 1984.

[8] M. Farmen and J. S. Marron. An Assesment of Finite Sample Performace of Adaptive Methods inDensity Estimation. In *Computational Statistics and Data Analysis*, 1998.

[9] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical Inference and Data Mining. *Communications of the ACM*, 39(11), November 1996.

[10] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In B. Yormark, editor, *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pages 47–57. ACM Press, 1984.

[11] A. Henrich. The LSD^h-Tree: An Access Structure for Feature Vectors. In *Proceedings of the Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*, pages 362–369. IEEE Computer Society, 1998.

[12] R. Jackendoff. The architecture of the linguistic-spatial interface. In P. Bloom, M. A. Peterson, L. Nadel, and M. F. Garrett, editors, *Language and Space*, pages 1–30. MIT Press, 1996.

[13] N. Katayama and S. Satoh. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In J. Peckham, editor, *Proceedings ACM SIGMOD 1997 International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 369–380. ACM Press, 1997.

[14] V. Katkovnik and I. Shmulevich. Nonparametric Density Estimation with Adaptive Varying Window Size. In *Conference on Image and Signal Processing for Remote Sensing VI, European Symposium on Remote Sensing, Barcelona, Spain, September 25-29*, 2000.

[15] D. A. Keim. Visual data mining. tutorial notes, Athens, Greece 1997.

[16] D. A. Keim and H.-P. Kriegel. Visdb: A system for visualizing large databases. In M. J. Carey and D. A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995*, page 482. ACM Press, 1995.

[17] J. Lee, D. Kim, and C. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In A. Delis and C. Faloutsos and S. Ghandeharizadeh, editor, *Proceedings of the ACM SIGMOD 1999 International Conference on Management of Data, June, Philadephia, Pennsylvania, USA*, pages 205–214. ACM Press, 1999.

[18] W. Lorensen and H. Cline. Marchine cubes: A high resolution 3d surface construction algorithm, 1987.

[19] A. Mažeika, M. Böhlen, and P. Mylov. Density Surfaces for Immersive Explorative Data Analyses. Proceedings of the Workshop on Visual Data Mining, in conjunction with Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001.

[20] H. R. Nagel, E. Granum, and P. Musaeus. Methods for Visual Mining of Data in Virtual Reality. Proceedings of the International Workshop on Visual Data Mining, in conjunction with ECML/PKDD2001 - 2th European Conference on Machine Learning (ECML'01) and 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01), 2001.

[21] J. T. Robinson. The K-D-B-Tree: A Search Structure For Large Multidimensional Dynamic Indexes. In Y. Edmund Lien, editor, *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data, Ann Arbor, Michigan, April 29 - May 1, 1981*, pages 10–18. ACM Press, 1981.

[22] S. Sain. Adaptive Kernel Density Estimation, 1994.

[23] S. Šaltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the Positions of Continuously Moving Objects. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, volume 29, pages 331–342. ACM, 2000.

[24] N. Sawant, C. Scharver, J. Leigh, A. Johnson, G. Reinhart, E. Creel, S. Batchu, S. Bailey, and R. Grossman. The Tele-Immersive Data Explorer: A Distributed Architecture for Collaborative Interactive Visualization of Large Data-sets. In *Proceedings of the Fourth International Immersive Projection Technology Workshop*, Ames, June 2000.

[25] B. J. Scholl. Objects and attention: the state of the art. *Cognition*, 80:1–46, 2001.

[26] D. W. Scot. *Multivariate Density Estimation*. Wiley & Sons, New York, 1992.

[27] H. Shen and C. Johnson. Sweeping Simplicies: A Fast Isosurface Extraction Algorithm for Unstructured Grids. In *Proceedings of Visualization 1995*, pages 143–150, Atlanta, GA, October 1995. IEEE Computer Society Press.

[28] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London, 1986.

[29] C. C. Taylor. Bootstrap Choice of the Smoothing Parameter in Kernel Density Estimation. *Biometrika*, 76(4):705–12, 1989.

[30] S. Ullman. High-level Vision. Object Recognition and Visual Cognition. *Cambridge, Massachusetts, The MIT Press*, 1996.

[31] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman & Hall, London, 1985.

[32] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[33] E. J. Wegman and Q. Luo. Visualizing Densities. Technical Report Report No. 100, Center for Computational Statistics, George Mason University, 1994.

[34] D. A. White and R. Jain. Similarity Indexing with the SS-tree. In S. Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 516–523. IEEE Computer Society, 1996.

[35] J. Wilhelms and A. Van Gelder. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.